Seeking Efficiency for the Accurate Draping of Digital Garments in Production

José M. Pizana, Gabriel Cirio, Alicia Nicas, and Alejandro Rodríguez

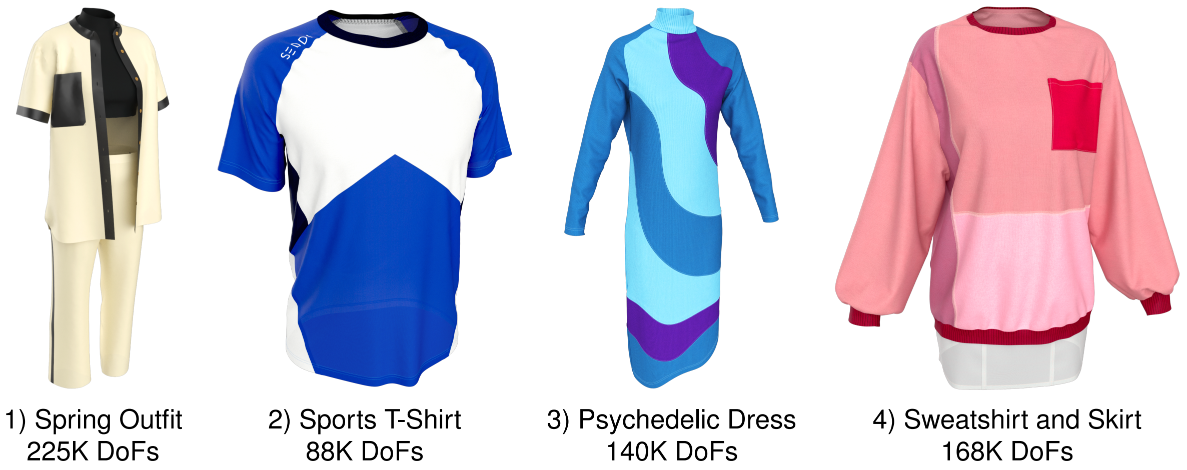


Fig. 1: Four of the production garments used to benchmark the different features presented in this paper. Together with the six other garments of Figure 4, they represent a wide range of possible real-world patterns, with varying constructions and complexity, all used in a production environment.

Abstract— Digital garments are set to revolutionize the apparel industry in the way we design, produce, market, sell and try-on real garments. But for digital garments to play a central role, from designer to consumer, they must be a faithful digital replica of their real counterpart: a digital twin. Yet, most industry-grade tools used in the apparel industry do not focus on accuracy, but rather on producing fast and plausible drapes for interactive editing and quick feedback, thus limiting the value and the potential of digital garments. The key to accuracy lies in using the proper underlying simulation technology, well documented in the academic literature but historically sidelined in the apparel industry in favor of simulation speed. In this paper, we describe our industry-grade cloth simulation engine, built with a strong focus on accuracy rather than sheer speed. Using a global integration scheme and adopting state of the art simulation practices from the Computer Graphics field, we evaluate a wide range of algorithms to improve its convergence and overall performance. We provide qualitative and quantitative insights on the cost and capabilities of each of these features, with the aim of giving valuable feedback and useful guidelines to practitioners seeking to implement an accurate and robust draping simulator.

Index Terms—Performance, Computer Animation, Simulation, Computer-aided Design

1 Introduction

The accurate simulation of digital garments opens exciting opportunities for the apparel industry. When simulated with enough fidelity, digital garments can play a central role during the design stage of a garment. Designers can have immediate feedback on their design choices, be it in patternmaking, sewing or fabric selection, by seeing how the digital garment drapes over a digital fit model. Digitalization frees the designer from building physical prototypes in the early stages of garment development, significantly reducing the development time, the cost and the ecological footprint of the process. At the other end of the development process, brands can leverage digital garments for high fidelity digital photography, and enable a return-free, personalized and compelling shopping experience by letting consumers try the digital garment through online virtual try on.

While the apparel industry is increasingly adopting digital garments, the adoption rate is surprisingly slow [5]. And even for those who have embraced digital transformation, the penetration is relatively shallow [27]: digitalization is often siloed to specific development stages, and existing solutions struggle to bridge the gap between design and production [52], limiting digital garments to a minor role.

Many engineering, manufacturing and even service industries, such as aerospace, automobile, electronics, architecture, etc, have largely embraced digitalization early on as an integral part of their design and development process. Why, then, are we not seeing a broader adoption of digital tools, and particularly digital twins, in the apparel industry?

One important reason is that garments are notably difficult to simulate accurately. Garments are highly deformable objects: their draping is centerpiece to the fit, the comfort, the functionality, and the appeal of a garment. Unfortunately, highly deformable objects are in general particularly hard to simulate, with collisions appearing in unpredictable places, strong nonlinearities stemming from large deviations from the rest pose, and a lack of clear linearization paths to simplify the problem. In addition, the variety of fabrics that can be used in a garment is virtually infinite, with mills producing tens of thousands of new textiles every year. Even seemingly subtle mechanical differences between two

-

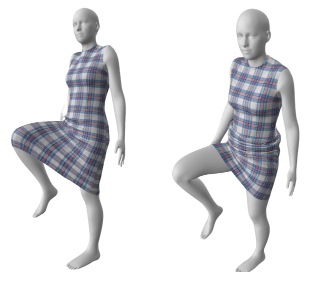
José M. Pizana, Gabriel Cirio, Alicia Nicas, and Alejandro Rodríguez are with SEDDI. E-mails in order of authorship: jmpizanagarcia@gmail.com, gabriel.cirio@gmail.com, alicia.nicasmiquel@gmail.com, alejandrora88@gmail.com

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

fabrics must be accurately captured by the simulation to fully depict a true digital twin without compromising the designer's intent. Numerically, dealing with the very high stretch but low bending stiffness of most fabrics, as well as their anisotropic and nonlinear behavior, is already a problem on its own. In addition, garments are complex assemblies of all sorts of objects beyond pieces of fabric, such as buttons, zippers, drawcords, sewing threads, etc, intertwined into one single but highly heterogeneous entity.

In addition to the complexity of accurately simulating a digital garment, and perhaps even due to it, existing solutions in the apparel industry trade off accuracy for speed. All garment simulation approaches in the academic literature tackle this inevitable trade off by favoring one or the other. For a given computational budget, if speed is prioritized then accuracy will degrade. The result is a fast and interactive simulation of the garment, which provides immediate visual feedback and lets the user make adjustments in real time. While useful on its own, this approach sacrifices accuracy for the sake of speed and a fully interactive experience: fast techniques use many of the approximations often found in videogame technology, far from the accuracy standards one would need in an engineering tool.

In the context of drape simulations, and of this work in particular, we understand accuracy as the capability of the digital *fabric* to stretch and bend like its real counterpart. This definition naturally extends to the garment as a whole, which is a combination of pieces of fabric sewn together. However, it is a purposely narrow definition: while fabric is arguably the predominant factor in the way a garment drapes, it is far from being the only one. Seams, trims, contact with the body and with itself, and soft body deformations are just a few examples of elements that can influence the final drape, and whose accuracy we exclude from our study for the sake of conciseness without dismissing their importance. With this context and definition in mind, we argue that the characteristic stiffness of real fabrics can challenge speed-oriented solvers, leading to insufficient convergence and therefore inaccuracy. Common manifestations of an inaccurate treatment are overly elastic behaviors [20], poor wrinkle formation [37] or more subtle, but also important artifacts, such as incorrect sliding treatment [38].



In the side figure, the avatar lifts his right leg, stretching the dress caught around the knee. For some real-time simulators with aggressive computational budget constraints, fabric can become unnaturally stretched (left). With an accurate simulator (right) that properly enforces fabric inextensibility, the fabric resists stretching and ends up sliding up the leg. Figure 13 and Appendix D show additional side by side examples to highlight the difference of accuracy between simulators.

We argue that this strong focus on real time simulation and fast draping results, with the resulting loss in accuracy, is what has prevented a more widespread use of digital garments in the apparel industry. A true, complete, in-depth adoption of digital garment tools can only be achieved if the digital garment is an accurate digital replica of the real garment: a digital twin. This way, digital garments can be used to make decisions from the earliest design stages, and all along its development lifecycle, bridging the gap between design, production and retail.

Accuracy starts by using the right simulation tools, the right simulation solver. There is an enormous body of research in the Computer Graphics field about the simulation of cloth, spanning almost 30 years. While initially limited to the visual effects and videogame industries, modern simulation techniques have the potential to produce highly accurate digital garments for apparel design as long as speed is not the driving factor. Unlike most existing solutions for the apparel industry, we want to embrace this category of techniques that prioritize accuracy.

Based on these observations, in this paper we describe how we have set up a cloth simulation system by adopting an approach that focuses on accuracy. However, an accurate but excruciatingly slow method is not the answer either, for the simple reason that it is not practical nor, in our case, commercially viable. We therefore leverage existing literature and propose some novel ideas to significantly speed up our simulator without compromising its accuracy, making it viable for the industry. We comprehensively study a wide set of convergence and performance features to assess their impact on computation cost and overall accuracy. By shedding light into the relative value of each feature, we aim to provide useful guidelines and informed advice to future practitioners seeking to implement a robust, accurate and efficient cloth simulator. The simulator has been proven since then with thousands of simulations of customer-made garments with real-life constructions and digitized fabrics.

2 Related Work

Cloth simulation has been an active research area for several decades. The study of cloth behavior is important for a wide range of fields, including textile development, apparel design, visual effects and video games. In this section, we first give an overview of the different simulation techniques that have been proposed to model cloth behavior, which can be categorized by their integration scheme. These are the backbone of cloth simulation in modern 3D CAD software for the apparel industry. We then survey existing production-grade cloth simulation packages, commercial or otherwise, making use of these simulation techniques, while highlighting their specificities in terms of accuracy and computation speed. We then focus on technical contributions that seek to provide, akin to our work, an in-depth understanding of specific cloth simulation engines with implementation guidelines and pitfalls to avoid.

2.1 Integration Schemes

Various numerical simulation methods have been proposed to solve the equations of motion governing the behavior of cloth. These methods can be categorized according to the way they solve the underlying system of equations, effectively acting as a trade-off between accuracy and speed: *global* methods and *local* methods.

Global methods rely on one or several linear solves to timestep an implicit (Backward Euler) discretization of the laws of motion. When the equations are highly nonlinear, which is often the case in cloth simulation with hyperelastic materials, Newton's method [48] is arguably the most accurate approach, properly handling the very high material stiffness found in most fabrics. Newton's method quadratic convergence is offset by the high computational cost of each iteration, which requires the computation of the energy's Hessian. Keeping the number of iterations low for both the linear solve (e.g., a Conjugate Gradient solver) and the Newton step without impairing the fidelity of the results is challenging, and we provide some guidelines for this in § 5.1. Some approaches take a single Newton step [2], effectively assembling and solving a linear system only once per time step and resorting to backtracking mechanisms if the step did not converge, but the cost of the iteration remains high. Recent methods re-formulate the problem as an energy minimization [21], allowing to use larger time steps. When fidelity can be sacrificed in favor of speed, several methods avoid computing the full energy Hessian matrix by using precomputed approximations and factorizations, effectively reducing the cost of setting up and solving the linear system while still benefiting from a *global* method. Some examples are Projective Dynamics [6], ADMM [50], and the Quasi-Newton L-BFGS update [37], potentially achieving interactive simulation speeds for some limited scenarios. Other approaches striving to increase speed while retaining fidelity include the multigrid method of Xian et al. [73], currently limited to homogeneous materials across the domain, and the subspace preconditioning approach of Li et al. [35], which heavily relies on GPU computation.

While *global* methods are interesting for their accuracy, they struggle to achieve interactive results for garments with even moderate mesh resolutions. Without enough time to converge, visual artifacts and instabilities quickly appear. This is when *local* methods become interesting, since they do not require assembling and solving large linear systems but rely on the fast initial convergence of the Gauss-Seidel method [3]. The seminal work by Müller et al. [44], Position Based Dynamics (PBD), models the material behavior using quadratic constraint formulations. The main shortcomings of PBD, namely the implicit coupling between material stiffness and the number of iterations as well as the size of the time step, is addressed in the follow up method Extended PBD (XPBD) [39]. Other methods have also built upon PBD to improve its convergence [46]. The stability and speed of these methods, together with their simplicity, have made them the go-to method for real-time simulation. The price to pay is in overall accuracy; even though XPBD is expected to converge to the true solution given enough iterations, it fails to converge in some stiff scenarios [38, 55].

2.2 Cloth Simulation Software

There is a wide variety of cloth simulation software packages built on top of *global* or *local* methods depending on specific needs. In the case of the apparel industry, and particularly in 3D CAD for apparel design, cloth simulation plays a central role in the early stages of garment design. A 3D CAD provides different tools for the patternmaker to draw, assemble and drape the garments in 3D, ideally delaying the need to build a physical prototype to a much later stage in the design process. Notable examples are Browzwear's VStitcher [10], Clo [18] and Optitex's PDS 3D [49]. All three provide cloth simulation capabilities to allow real-time interaction with the user and fast drapes for immediate visual feedback. As we focus on garment fidelity rather than sheer speed, in this paper we describe our cloth simulation engine which uses a *global* integration scheme.

When the process is artist driven, which is often the case for digital content creation for marketing and sales (digital photography, e-commerce), there are several generic 3D tools and simulation packages from the visual effects industry that allow cloth simulation with a strong emphasis on high visual quality rather than accuracy. Off-the-shelf cloth simulation tools, such as Autodesk Maya's nCloth [57] and SideFx's Houdini Vellum [54], overwhelmingly use *local* integration schemes, and with good reason: since the process is artistic, disconnected from real materials and focused on visuals, the fabric properties and garment drapes are usually hand-tuned to look good and can therefore sacrifice fidelity to a large extent.

Some industry-grade simulation engines reportedly use *global* schemes for cloth, but these are often in-house tools from visual effects companies not available to third parties [28, 31, 60]. However, the research teams behind them have published academic literature on some of the internal mechanisms of their engines, providing insightful guidelines, good practices and warnings against common pitfalls for practitioners wanting to implement robust, accurate and efficient global integration schemes. While these methods are not specific to cloth simulation, and target the visual effects industry in general, they are still valuable for the apparel industry. The most complete and thorough example is Kim and Eberle's course [28] about Pixar's simulation engine, Fizt2. The authors give an overview of the main algorithms under the hood for the simulation of deformable bodies, including cloth. Topics range from simulation performance to collision detection and response, including techniques for reducing the computational cost of the simulation while maintaining its accuracy, which is the main purpose of our work. They also provide detailed instructions on how to efficiently solve the equations of motion using a *global* integration scheme. Another example is the multiphysics approach of Weta Digital's Loki simulation engine [31], and particularly how it manages many different interoperating simulation systems and discretizations such as fluids, rigid bodies and deformable objects, all within a *global* integration scheme. In this paper, we describe how we leverage some of these insights in our own simulation engine.

Other commercial engines can be traced back to a seminal academic paper describing an innovative simulation method. This is the case of the multigrid cloth simulation work by Tamstorf et al. [60], later used in Disney's cloth simulation engine to achieve a one order of magnitude speedup in computation times, with seamless integration in a *global* solver. Nucleus, a unified solver based on soft constraints solved in an interleaved manner, is extensively described by Stam [57] while being the backbone of Autodesk Maya nCloth simulation. Similarly, Macklin et al. [40] introduced a unified particle-based approach, built on top of PBD, which would become NVidia Flex simulation toolkit. Unified solvers are interesting because they allow the simulation of cloth, rods, volumetric bodies, rigid bodies, and even fluids from a single common building block (particles, for example) and using a single solver which, unlike multiphysics approaches [31], greatly reduces the size and complexity of the system. Unfortunately, they often use *local* integration schemes, such as in both examples above. Han et al. [24] describe a production environment for cloth simulation based on PBD and making extensive use of the GPU for speeding up computations, in another example of a *local* integration scheme used for cloth simulation in the visual effects industry.

Finally, open-source projects are another way to learn about the inner workings of cloth simulation engines. The industry-grade toolkits Bullet [16] and Nvidia PhysX [15] have their source code available for anyone to see, but these are *local* solvers appropriate for real time interaction, not high fidelity. Open source *global* solvers can be found in academia, such as the now aging but still widely used ARCSim [47] adaptive cloth simulator, the Hobak [29] engine by Kim, which carried over the key lessons learned while developing Pixar's Fizt2 simulation engine, or C-IPC [34], a state of the art global method heavily focused on providing intersection-free collision guarantees.

3 Integrator Overview

Our main goal is to produce a garment's final drape as fast as possible without compromising accuracy. To this end we start off with a *global* method: the backbone of our simulator is a robust *Backward Euler-Newton* integrator assisted with Line Search. Algorithm 1 summarizes the main stages of the simulator. In the remainder of this paper, we describe and evaluate different acceleration techniques that help towards this objective. The focus of this work is exclusively on drape simulations and many of the features we describe in this paper are specifically tailored to that end, even though the simulator ultimately integrates the equations of motion.

3.1 Backward Euler-Newton Integrator

Given Newton's second law $\mathbf{F}(\mathbf{x},\mathbf{v}) = \mathbf{M}\dot{\mathbf{v}}$, $\mathbf{v} = \dot{\mathbf{x}}$, we can advance the system state $\mathbf{x}_0$, $\mathbf{v}_0$ at time t to the state $\mathbf{x}_s$, $\mathbf{v}_s$ at time $t+h$ using backward Euler discretization:

$$\begin{aligned}\mathbf{M}(\mathbf{v}_s - \mathbf{v}_0) &= h\mathbf{F}(\mathbf{x}_s, \mathbf{v}_s)\\ \mathbf{x}_s &= \mathbf{x}_0 + h\mathbf{v}_s\end{aligned}$$

which, after substitution yields the nonlinear equation

$$\mathbf{f}(\mathbf{v}_s) = \mathbf{M}(\mathbf{v}_s - \mathbf{v}_0) - h\mathbf{F}(\mathbf{x}_0 + h\mathbf{v}_s, \mathbf{v}_s). \tag{1}$$

Given an initial guess $\mathbf{v}_i$ (we always set it to $\mathbf{v}_0$), Newton's method finds $\mathbf{f}(\mathbf{v}_{i+1}) = 0$ by iteratively refining the solution

$$\mathbf{v}_{i+1} = \mathbf{v}_i - \left(\frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{v}_i)\right)^{-1} \mathbf{f}(\mathbf{v}_i) \tag{2}$$

with

$$\frac{\partial \mathbf{f}}{\partial \mathbf{v}} = \mathbf{M} - h^2 \frac{\partial \mathbf{F}}{\partial \mathbf{x}} - h\frac{\partial \mathbf{F}}{\partial \mathbf{v}}. \tag{3}$$

We substitute (1) and (3) into (2) and rearrange the terms to get

$$\left(\mathbf{M} - h^2 \frac{\partial \mathbf{F}_i}{\partial \mathbf{x}} - h\frac{\partial \mathbf{F}_i}{\partial \mathbf{v}}\right)\Delta\mathbf{v} = \mathbf{M}(\mathbf{v}_0 - \mathbf{v}_i) + h\mathbf{F}_i \tag{4}$$

with $\mathbf{F}_i = \mathbf{F}(\mathbf{x}_0 + h\mathbf{v}_i, \mathbf{v}_i)$ and $\Delta\mathbf{v} = \mathbf{v}_{i+1} - \mathbf{v}_i$. Newton's method iterates until the error $r_n = \|\mathbf{f}(\mathbf{v}_{i+1})\| < \tau_n$ for a given threshold τ_n or until early stop (discussed in § 3.1.3). Note that the error in the step $i+1$ is conveniently the norm of the right-hand side: $\|\mathbf{M}(\mathbf{v}_0 - \mathbf{v}_{i+1}) + h\mathbf{F}_{i+1}\|$. At convergence, we set $\mathbf{v}_s = \mathbf{v}_{i+1}$ and $\mathbf{x}_s = \mathbf{x}_0 + h\mathbf{v}_s$.

Algorithm 1 Overview of the simulator's integration step. *objs* are the simulation objects and **A** and **b** refer to the left- and right-hand sides of Equation 4. The sections corresponding to each feature appear after comment markers (//).

```
1  initialParameterization($objs$,$\tau_{cg}$,$\tau_n$,$\omega$)          // § 5.1, 5.3
2  precomputations($objs$)                              // § 6.1, 6.4, 6.5
3  $h \leftarrow h_{max}$
4  $t \leftarrow 0$
5  while $t < t_{sim}$ do
6      initialRelaxation($t$, $h$, $objs$)                    // § 5.2
7      $c \leftarrow$ detectCollisions($objs$)
8      $\mathbf{x}_0$,$\mathbf{v}_0 \leftarrow$ getState($objs$)
9      $\mathbf{v}_i \leftarrow \mathbf{v}_0$,  $\mathbf{x}_i \leftarrow \mathbf{x}_0 + h\mathbf{v}_0$
10     $\mathbf{b} \leftarrow$ computeRhs($objs$, $c$, $\mathbf{x}_i$, $\mathbf{v}_i$)
11     $r_n \leftarrow ||\mathbf{b}||$
12     for $i_n = 0$ to $I_n$ do
13         $\mathbf{A} \leftarrow$ computeLhs($objs$, $c$, $\mathbf{x}_i$, $\mathbf{v}_i$)
14         $\Delta\mathbf{v} \leftarrow$ linearSolve($\mathbf{A}$, $\mathbf{b}$, $\tau_{cg}$)         // § 6.3, 6.2
15         $f_{ls} \leftarrow 1$
16         $diverged \leftarrow$ true
17         for $i_{ls} = 0$ to $I_{ls}$ do
18             $\mathbf{v}_{ls} \leftarrow \mathbf{v}_i + f_{ls}\Delta\mathbf{v}$,  $\mathbf{x}_{ls} \leftarrow \mathbf{x}_0 + h\mathbf{v}_{ls}$
19             $\mathbf{b} \leftarrow$ computeRhs($objs$, $c$, $\mathbf{x}_{ls}$, $\mathbf{v}_{ls}$)
20             if $||\mathbf{b}|| < r_n$ then
21                 $r_n \leftarrow ||\mathbf{b}||$
22                 $diverged \leftarrow$ false
23                 $\mathbf{v}_i \leftarrow \mathbf{v}_{ls}$,  $\mathbf{x}_i \leftarrow \mathbf{x}_{ls}$
24                 break
25             end
26             $f_{ls} \leftarrow 0.5 f_{ls}$
27         end
28         if $diverged$ then
29             break
30         end
31         if $r_n < \tau_n$ then
32             $t \leftarrow t + h$
33             setState($objs$, $\mathbf{x}_i$, $\mathbf{v}_i$)
34             break
35         end
36     end
37     if $checkEarlyStop(objs)$ then                      // § 5.4
38         break
39     end
40     updateTimestep($h$, $\omega$, $diverged$)
41 end
```

The left-hand side matrix of the system has to be at least positive semidefinite to guarantee Newton's method convergence. In practice, we have seen that our simulator is robust to energies that do not guarantee such matrices. This is mainly due to a time splitting scheme that helps maintain the positive semidefiniteness of the system as we discuss in § 3.1.3.

3.1.1 Conjugate Gradient

Our integrator uses a Preconditioned Conjugate Gradient (PCG) as the iterative method for solving the linear system of equations (line 14 in Algorithm 1).

Modified versions of the algorithm have been presented to efficiently handle constrained particles in the context of cloth simulation such as MPCG [2] and PPCG [60]. In our case, collisions are treated as soft constraints instead of hard constraints, so we can use the Eigen package [23] implementation of the PCG algorithm, which iterates until $r_{cg} < \tau_{cg}$, with $r_{cg} = \frac{|\mathbf{Ax}-\mathbf{b}|}{|\mathbf{b}|}$.

3.1.2 Line Search

Newton's method converges rapidly towards the solution when the current estimate is close enough to the solution. However, in some non-linear settings the step of a Newton iteration could point in the right direction but miss the step length [48]. Adding a line search scheme has been shown to increase robustness while adding a fraction of the cost to each Newton iteration [21, 26, 41].

We implement a backtracking line search approach similar to [26], consecutively halving the Newton step length until r_n decreases (lines 17-27 in Algorithm 1).

3.1.3 Time Splitting

Despite the use of a line search strategy, a dynamic simulation can reach challenging states (collision-rich scenarios, fast-moving fine wrinkles, etc.) that prevent a fast convergence of Newton's method, requiring many iterations to solve. The stiffness and damping matrices can even become non-positive definite in some states, leading to non-positive definite system matrix and to a divergence of the iterative method.

Analyzing the terms of the system matrix in (4), a decrease in h reduces the contribution of the stiffness ($\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}}$) and damping ($\frac{\partial \mathbf{F}_i}{\partial \mathbf{v}}$) matrices. As a result, it increases the dominance of the mass matrix $\mathbf{M}$ which, in our case, is lumped and therefore constant and positive definite, thus helping maintain the overall health of the system matrix. Adaptive time-stepping strategies can benefit from this property [2].

We also exploit this trait and implement a time-splitting scheme to deal with these challenging states (line 40 in Algorithm 1). We check if the step does not succeed after a fixed number of Newton iterations or if the residual increases after any Newton iteration. When any of these checks is triggered, the current simulation step aborts, resetting the state of the simulation objects to their state before the step and setting $h = \omega h$ with the time-splitting factor $0 < \omega < 1$ to integrate again.

Challenging configurations are typically transient, so the splitting is reversed by setting $h = \omega^{-1} h$ after several consecutive successful simulation steps while $h < h_{max}$. We set $h_{max} = 1ms$, as we explain in § 3.3.

3.2 Energies

Cloth objects are discretized as Delaunay triangle meshes and governed by elastic and damping energies. We use an anisotropic Saint Venant-Kirchoff (StVK) model [67] and a discrete hinge-based bending model [8, 22, 59] to capture the anisotropic stretch, shear and bending properties of cloth. We note that these models neglect some known behaviors of cloth, such as plastic deformation due to internal friction [43]. However, we have validated that after properly fitting the model parameters, they faithfully replicate the draping behavior of the different materials present in the garment (Figure 2). We use dissipation potentials [58] as damping energies, which we further discuss in § 5.3. Garment trims such as buttons are simulated as rigid bodies with spring attachments to the fabric.

We use standard point repulsion with quadratic energy penalty potentials [42] together with Coulomb friction [75] for all the collisions detected among objects in the scene (§ 3.3). We use a stiffness of $10^2 N \cdot m$ for all collisions.

Parallelization is easy to apply among energy stencils, with the caveat that synchronization is necessary because the degrees of freedom of different stencils can partially overlap. Our baseline uses this approach, and we develop this further in § 6.5.

3.3 Collision Detection

The interactions between objects play a major role when simulating complex scenes. Properly handling collision detection and collision resolution is therefore paramount.

We use discrete collision detection for interactions between all dynamic and static objects in the scene. We rely on acceleration structures for efficiently querying and discovering contacts in each step: we use distance fields [45] for static objects such as avatars and garment trims, and use bounding volume hierarchies to efficiently perform vertex-triangle and edge-edge proximity queries among cloth objects.

Implementing a robust collision detection and resolution system, particularly among codimensional objects such as cloth, is known to be (and remains) a very challenging problem. This has spanned research on the use of continuous collision detection (CCD) to prevent invalid configurations in every step [7, 25, 33, 62, 63]. Implementing a robust

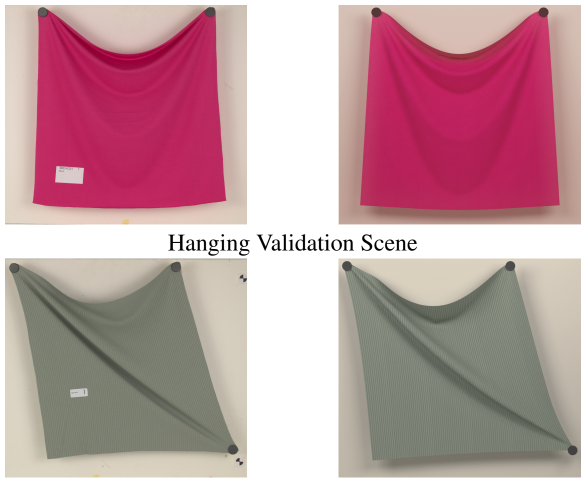


Fig. 2: Two validation scenes used to visually assess, in a controlled environment, the accuracy of a digital fabric compared to a real photograph. On the top pair, fabric hangs from two points and drapes under gravity. On the bottom pair, fabric is also stretched from its bottom right corner to produce buckling interactions.

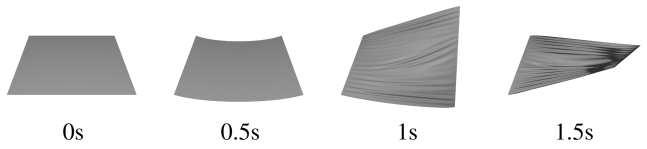


Fig. 3: Snapshots of the progression of a Synthetic Scene test for the (Fine Stiff) set of parameters τ_n =1e-3, τ_{cg} = 1e-2, and ω = 0.75.

CCD system, however, can be particularly challenging [61, 68, 70]. If not done carefully, it can easily become the bottleneck of the simulator. The recent IPC method [33, 34] offers extremely robust handling of collisions, but these guarantees come at the expense of performance as we will show in section § 7.2. In addition, in a production environment where there is no absolute control over the input to the simulator, additional systems such as untangling strategies tightly coupled with the collision system are required [28].

Discrete collision detection is much simpler to implement and faster to compute but does not offer theoretical guarantees and may not be suitable for all types of scenes. However, we have found that in the context of drape simulations, the use of discrete collision detection together with setting $h_{max} = 1ms$ to avoid large displacements and an initial relaxation (§ 5.2) to better condition the input, leads to very robust detection in practice. Other approaches in this direction are also proposed in the literature, such as limiting the maximum per-vertex displacement per step to prevent missing collisions [56], but we have found our approach to work well in all reasonable scenarios. In section § 7.2 we compare our approach with the C-IPC method in draping scenarios, showing good accuracy in the treatment of collisions, while providing a much better performance.

4 Experimental Setup

In the remainder of the paper, we focus on different features to accelerate drape computation without sacrificing quality. In this section we describe the experimental setup used to evaluate and discuss each of these features.

4.1 Methodology

We divide the set of features into two categories: convergence features (§ 5) and performance features (§ 6). Convergence features achieve drape speedups by changing the behavior of the system. Each feature is tailor-made for the specific problem of simulating drapes. On the other hand, performance features optimize the algorithms used to solve the system. They do not require specific knowledge on physically based cloth simulation and are therefore plug-and-play accelerations. This separation allows us to reduce the mutual influence across features to better study them in isolation. While all features are independent (each can be turned on or off at will), there are some interesting synergies that will be discussed when appropriate.

For each feature, we run a set of simulations with it turned on, which are compared against baseline simulations to understand the qualitative and quantitative effect of its activation. Unless stated otherwise, the simulations run until 3s of simulation time have elapsed. We repeat each test 3 times and report the averaged results. We store all the appropriate simulation data and analyze it after having run all simulations. Data sampling and saving costs are excluded from all reported time measurements.

4.2 Apparatus

We test each feature on a set of 10 production garments and use synthetics scenes for range sweeps during parameter tuning to reduce computations. All the experiments are run on a system with an Intel Core i7-7700K CPU with 64GB of memory.

4.2.1 Full Garments

To evaluate the effect of each feature, we selected a set of 10 production garments with a wide range of shapes, styles and seams, ranging from a plain t-shirt to a multilayer, buttoned up outfit. As in most CADs for the apparel industry, the garments were assembled using a real-time, interactive and low-resolution simulator. In particular, we used a Position-Based Dynamics solver. Figures 1 and 4 show all 10 garments.

The many fabrics making up these garments were digitized in a laboratory using specific fabric capture equipment, which includes a Uniaxial Stretch Test device for stretch parameters and a Pearloop Test device for bending parameters. The captured data is then used to derive mechanical parameters in weft, warp and bias directions for stretch and bending via the numerical optimization of test simulations. The resulting materials correctly reflect the mechanical behavior of the real fabrics, as shown in Figure 2 for two fabrics. As such, they are usually very stiff in stretch and very compliant in bending, a challenging situation for any accurate simulator. Appendix B in the supplementary material shows the mechanical parameters of each fabric used in these garments.

4.2.2 Synthetic Scenes

Some features, such as in § 5.1, require many tests to be able to draw any conclusion. To this end, we have devised a synthetic scene that is faster to simulate than the 10 production garments while triggering a larger range of deformation modes.

The synthetic scene consists of a flat, squared 1m^2 piece of fabric clamped at two opposing sides. After 0.5s of simulation, one of the clamped sides rotates during 1s until reaching 0.75π degrees, producing stretching, bending and self-collision forces. The simulation keeps running until it reaches 3s. Some snapshots of this process are shown in Figure 3. We use two possible discretizations (a coarse triangulation with 30K DoFs and a fine triangulation with 300K DoFs) and two sets of materials (a stiff woven fabric and an elastic knitted fabric). Appendix B in the supplementary material shows the mechanical parameters of each fabric used in these scenes.

5 Convergence Features

The first set of features that we evaluate share the common property of significantly affecting the dynamics and convergence of the system, such as choosing the error thresholds of the integrator or deciding when draping simulations should stop. In this section we describe the set of features and study their effect.

5.1 Solver Parameters

Our solver, like most, has a set of parameters that control its numerical behavior. Two major parameters are the τ_n and τ_{cg} thresholds respectively required by the Newton (§ 3.1) and the Conjugate Gradient

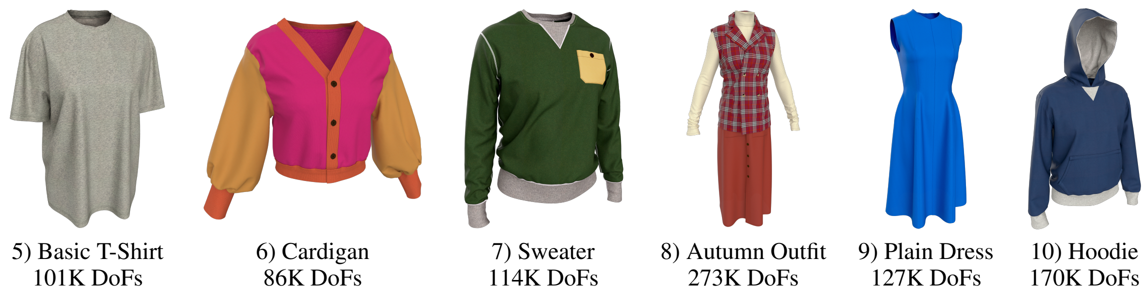


Fig. 4: Six additional production garments used throughout this paper, showing a range of shapes, sizes and styles, including tailoring features such as buttons, overlapping layers, pockets and facings.

(§ 3.1.1) methods to define their stop criteria. Values for these thresholds are typically only hinted at in the literature for arbitrary setups or not reported at all and left for the user to guess. However, they can have a significant impact on the performance and accuracy of the solver. We therefore study the quantitative and qualitative implications of choosing different τ_n and τ_{cg} values, and we show which values best fit the goal of obtaining a drape.

Another relevant parameter is the time-splitting factor ω described in § 3.1.3. The ideal (but difficult to reach) state would be to always have $h = h^*$, being h^* the highest value that allows successful integration steps. A low ω can rapidly reach a successful h value but stay far from h^*, therefore advancing at sub-optimal speed. A high ω can reach a value closer to h^* but may require multiple splitting operations, wasting time on retries when decreasing h and on small incremental changes when increasing h. We therefore also explore different choices of ω to understand their effect on overall performance.

There are other parameters that we do not explore in this work. In particular, we set $h_{max} = 1ms$ to increase the robustness of our discrete collision detection system as explained in § 3.1.3. We also set a maximum of 500 iterations for the Conjugate Gradient solver, as we sometimes observe asymptotically decreasing residuals that lead to costly linear solves. With this maximum we prevent them and let the rest of the robustness strategies take over. Similarly, we fix the number of maximum line search iterations I_{ls}, the number of maximum Newton iterations I_n and the number of consecutive successful steps I_s before increasing h, all of which we set to 5 as we have found them to be of little influence in our context as long as they remain small. We note, however, that they could become relevant under other circumstances, for example, if larger time steps were to be taken [21].

5.1.1 Broad Analysis

We start with a wide sweep of τ_n from 10^{-7} to 10^{-1} to later narrow the search to the relevant range. We also test $\omega \in [0.25, 0.5, 0.75]$ and fix $\tau_{cg} = 10^{-2}$. We use the synthetic scenes, as they exhibit the most significant effects found in production (stretch, wrinkling and collisions) while also (sparsely) covering typical materials and triangle counts found in production garments. We later confirm that they serve as a good proxy for production scenes.

Since our simulation includes Coulomb friction, slightly different equilibrium drapes are expected for different parameter configurations. We analyze the resulting geometry to identify large differences and/or visual artifacts in the final drapes. We measure the average vertex distance of the resulting drapes against the $\tau_n = 10^{-7}, \omega = 0.5$ case. Table 1 shows the results after running simulations for all the combinations, reporting total computation time and average vertex distance.

The experiments show that τ_n with values between 10^{-3} and 10^{-1} perform significantly better than in the 10^{-7} to 10^{-4} bracket, with a substantial performance improvement up to 6x. For the less restrictive τ_n range, the error in drape similarity increases but remains bounded within an maximum average vertex distance of 12.23mm. The results also reflect that the choice of ω is not relevant when the time-splitting approach does not trigger often, but when it does $\omega = 0.75$ leads to a better performance.

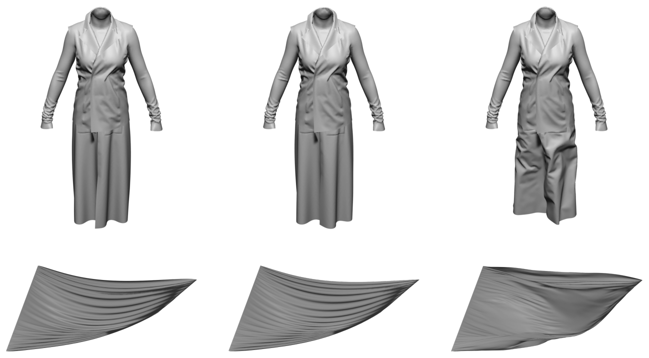


Fig. 5: Three drapes of the same garment (top row) and the Fine & Soft synthetic scene (bottom row). The ground truth ($\tau_{cg} = 10^{-3}$) is shown in the center column. On the left, the scenes are successfully simulated with ($\tau_{cg} = 10^{-2}$), reaching convergence. On the right, the scenes simulated with ($\tau_{cg} = 10^{-1}$), diverge due to energy injections.

Note: exiting the Newton method too early is known to introduce numerical damping [65], and we are likely experiencing varying degrees of this effect when changing τ_n. However, since our goal is the final drape, we can accommodate (and even welcome) some additional damping as long as the drape remains accurate and the performance gain out-weights the slower dynamics, improving wall-clock simulation times. Conclusions would very likely be different if the goal involved animation or accurate trajectories, where artificial damping is often undesirable.

5.1.2 Narrow Analysis

Based on the previous results, we set $\omega = 0.75$ and do a new evaluation of τ_n ranging from 10^{-5} to 10^{-1} and add $\tau_{cg} \in [10^{-3}, 10^{-2}, 10^{-1}]$ to the analysis. We use [$\tau_n = 10^{-7}$, $\tau_{cg} = 10^{-2}$, $\omega = 0.75$] as ground truth, as we have it from the previous analysis. We perform this new evaluation on both production and synthetic scenes. The results are summarized in Table 2 for the synthetic scenes.

Synthetic scenes: We observe only marginal performance gains from using τ_n values less restrictive than 10^{-3}. Closer inspection shows that there is barely any h adjustment in the $\tau_n \in [10^{-3}, 10^{-2}, 10^{-1}]$ range for these scenes, and multiple Newton iterations are required only in the more challenging collision configurations during the torsion. A less restrictive τ_{cg} does lead to a performance gain due to lower number of CG iterations. However, for $\tau_{cg} = 10^{-1}$ we start to see divergent behavior in the form of energy injections and penetrations in both the soft coarse and soft fine cases, as reflected by the corresponding vertex distances of Table 2, and as shown in Figure 5 (bottom row).

		Coarse & Soft			Coarse & Stiff			Fine & Soft			Fine & Stiff		
	τ_n \ ω	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
Comp. Time (s)	10^{-7}	835	824	871	1155	791	759	14144	11176	11181	32203	19892	21003
	10^{-6}	621	608	607	759	564	551	6958	6916	6803	12054	11832	12130
	10^{-5}	499	474	474	354	355	357	5273	5421	5361	11538	8069	7758
	10^{-4}	366	367	371	335	335	340	4099	4109	4134	11382	7616	5465
	10^{-3}	306	308	309	324	324	327	3548	3564	3590	5295	5242	5322
	10^{-2}	283	284	286	305	305	310	3312	3349	3347	4898	4987	5003
	10^{-1}	282	283	287	305	306	312	3301	3325	3370	4897	4960	4971
Mesh Distance (mm)	10^{-7}	0.21	0.0	0.09	0.12	0.0	0.02	3.41	0.0	1.12	0.79	0.0	1.05
	10^{-6}	2.05	2.38	2.49	0.12	0.01	0.06	3.77	3.79	3.76	2.39	2.4	2.41
	10^{-5}	2.83	2.82	3.24	0.16	0.16	0.16	4.17	4.42	4.39	2.41	1.66	1.17
	10^{-4}	3.77	3.59	4.01	0.25	0.23	0.22	4.54	5.26	5.64	2.5	5.55	4.45
	10^{-3}	3.33	4.15	5.51	0.29	0.26	0.3	9.83	10.73	10.51	4.37	4.42	4.48
	10^{-2}	4.49	4.63	3.75	0.81	0.84	0.82	12.23	10.76	11.6	1.07	1.07	1.07
	10^{-1}	2.72	3.27	4.31	0.81	0.84	0.82	11.07	12.11	11.64	1.07	1.07	1.07

Table 1: Results for the broad analysis of solver parameters, varying the Newton threshold τ_n between 10^{-7} and 10^{-1} and the time splitting factor ω between 0.25 and 0.75. Overall, a less restrictive τ_n gives the best results without significant accuracy loss, and the choice of ω is relevant only when significant time splitting occurs.

		Coarse & Soft			Coarse & Stiff			Fine & Soft			Fine & Stiff		
	τ_n \ τ_{cg}	10^{-3}	10^{-2}	10^{-1}	10^{-3}	10^{-2}	10^{-1}	10^{-3}	10^{-2}	10^{-1}	10^{-3}	10^{-2}	10^{-1}
Comp. Time (s)	10^{-5}	459	449	541	370	341	377	5514	5069	6852	8726	7409	6109
	10^{-4}	362	357	379	367	326	307	4256	3961	4142	6406	5208	4216
	10^{-3}	315	314	315	336	340	308	3752	3594	3598	6390	5406	4416
	10^{-2}	306	305	307	337	326	308	3655	3520	3356	6437	5422	4222
	10^{-1}	305	305	297	336	328	316	3663	3529	3404	6430	5356	4270
Mesh Distance (mm)	10^{-5}	1.75	3.17	5.5	0.3	0.3	0.32	4.29	4.73	7.23	1.28	0.91	0.9
	10^{-4}	1.54	3.5	12.21	0.3	0.3	0.38	4.03	4.53	11.03	1.83	4.34	8.76
	10^{-3}	3.23	3.85	14.1	0.31	0.25	0.88	4.01	9.96	37.94	4.66	4.49	9.5
	10^{-2}	3.88	2.76	32.61	0.31	0.29	0.93	3.8	10.85	48.15	6.6	1.07	5.83
	10^{-1}	3.83	4.33	30.78	0.31	0.29	0.93	3.71	12.37	59.03	6.6	1.07	5.86

Table 2: Results for the narrow analysis of solver parameters, varying the Newton threshold τ_n between 10^{-5} and 10^{-1} and the Conjugate Gradient threshold τ_{cg} between 10^{-3} and 10^{-1}. Overall, a less restrictive τ_n gives the best results without a significant loss in accuracy. τ_{cg} also decreases computation times but at the cost of a significant loss in accuracy in Soft cases, which is exacerbated for the less restrictive values of τ_n.

Production scenes: Similar conclusions can be drawn for the production scenes. The τ_n variation in the $[10^{-3}, 10^{-1}]$ range has a very limited impact, since all the simulations do a single Newton iteration per step and no time-splitting. However, it performs faster than the $\tau_n = 10^{-5}$ ground truth while reaching a similar drape. As with the synthetic scenes, $\tau_{cg} = 10^{-1}$ leads to faster computation, but visual inspection shows energy injections for some of the scenes which are not present for $\tau_{cg} = 10^{-2}$, as shown in Figure 5 (top row).

In light of this analysis, we select $\tau_n = 10^{-3}$, $\tau_{cg} = 10^{-2}$, $\omega = 0.75$ and use them for the rest of the experiments in this work. Naturally, this process would need to be repeated and new values would need to be selected if any of the fixed parameters (e.g., h_{max}) were to change in the future, given that all the solver parameters are coupled to some extent. However, these experiments show that fine-tuning integrator parameters can provide a significant performance speedup with little deviation from the ground truth. In addition, using a reduced set of synthetic scenes that sample the production space serves well to help in the search for optimal values.

5.2 Initial Relaxation

The first few time steps of a simulation are particularly sensitive to the initial conditions of the problem. A badly conditioned input mesh can introduce a considerable amount of error from which the simulation might not recover. This can happen when the input mesh comes, for example, from an artistic pipeline that does not consider real fabric properties, or from an interactive design tool that gives an initial approximation of the assembled garment [36]. The input mesh can contain a very high potential elastic energy that, together with the typically stiff stretching behavior of fabrics, can lead to challenging high-speed, contact-rich dynamics, compromising the entire simulation.

This has led researchers to devise different relaxation schemes, such as reducing the high initial potential energies [76] or reaching a feasible initial state [77]. We address this issue by setting an initial time step $ht = ht_{max}/16$ and defining a relaxation window spanning the first 0.1s of simulation. During this window, velocities are set to 0 after each integration step and fabric stiffness values are multiplied by a factor linearly interpolated in the range $[10^{-2}, 1]$ over the relaxation window duration.

While this strategy is simple, we have found it to work well in practice for all scenes, preventing otherwise incorrect simulations such as the one shown in Figure 6 after a sloppy pinching operation. For scenes with a good initial state, such as the 10 production scenes, the additional computation time induced by the relaxation strategy is negligible. We have observed that, even in these cases, our early stop strategy (described in § 5.4) can sometimes benefit from this reduced initial energy state.

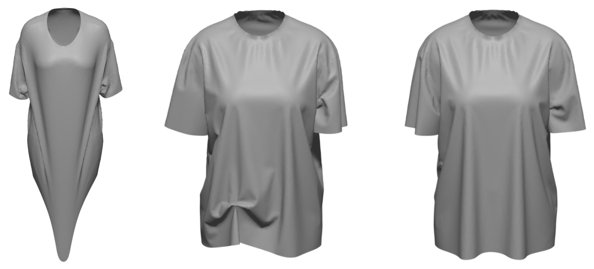


Fig. 6: A badly initialized garment (left) leads to a very high potential energy state at the start of the simulation. Attempting a regular simulation of this garment (center) leads to an undesirable configuration, while the use of our initial relaxation approach (right) successfully reaches the expected drape.

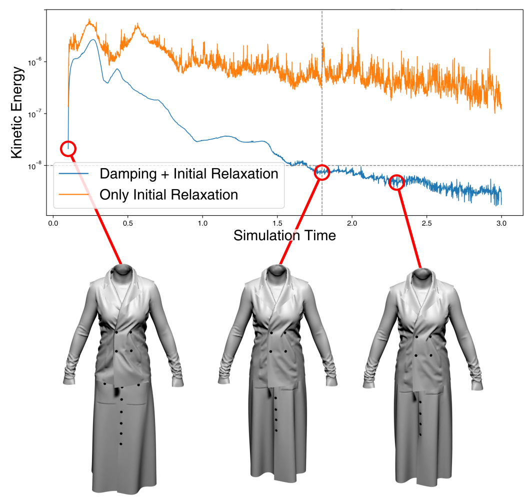


Fig. 7: Damping effect on a garment's kinetic energy. Without damping, kinetic energy remains high and the simulation does not reach the early stop threshold. The visual difference between the early stop timestamp (bottom center) and 0.5s later (bottom right) is negligible, validating the decision of stopping early.

5.3 Near-Critical Damping

Real world objects dissipate kinetic energy due to multiple reasons. While the ubiquitous Rayleigh damping [51] is still used in many engineering contexts, different damping models have been proposed to simulate these dissipative processes in controlled ways [2, 9, 58, 74].

To dissipate general system velocity, we add a simple air drag force opposing a vertex with mass m_i and velocity $\mathbf{v}_i$:

$$\mathbf{F}_{d,i}(\mathbf{v}_i) = -\alpha m_i \mathbf{v}_i \tag{5}$$

with damping coefficient α.

To dissipate specific potential energies, we follow Sánchez-Banderas model [58], defining dissipation potentials for our stretch and bending energies based on their strain rate $\dot{\varepsilon} = \nabla_{\mathbf{x}} \varepsilon^T \mathbf{v}$. For stretch, Sánchez-Banderas provides the expressions of the dissipative forces and Hessians of the Saint Venant-Kirchoff energy. For bending, we derived the expressions following the same approach. We refer the reader to Appendix A of the supplementary material for more details.

It is hard to come by generic damping parameters that would work across the board in all scenarios and for all fabrics. Damping parameters typically require careful tuning to match a desired behavior [74]. In our case, the three parameters α, β_s and β_b control the air drag, stretch and bending damping forces respectively. Since we are interested in final drapes, we set our coefficients pushing past the realistic dynamic behavior, and rather approaching a near critically damped system to shorten the simulation time required to reach equilibrium. We have empirically evaluated which values draw near this behavior for the scales and material ranges that we typically find in production scenes, reaching $\alpha = 5s^{-1}$, $\beta_s = 10^{-2}$ and $\beta_b = 10^{-1}$.

Figure 7 shows a comparison with and without these damping forces. Their addition, as expected, increases the overall simulation cost as they slightly worsen the conditioning of the system matrix as well as require additional force and Hessian accumulations. However, this is largely offset by a significant reduction in kinetic energy, allowing the system to converge to equilibrium faster. This particularly benefits our early stop criterion described in the following section which, taken together, significantly reduces computation times.

5.4 Early Stop Criterion

Our simulations are expected to run until reaching (near) equilibrium. Simply setting a generous simulation time would suffice to reach this equilibrium at some point, but at the expense of pointless, wasted computation cycles. Instead, we define a criterion to trigger an early stop when reaching a target equilibrium state.

One possible early stop criterion could be setting a threshold to the total system kinetic energy. However, this approach has two issues. First, the total kinetic energy of a system with many degrees of freedom does not allow to discern when the whole system is near rest and when most of the system is at rest except for a small region still undergoing dynamics, making it virtually impossible to find a threshold that satisfies all cases. Second, the system can go through transient low kinetic states to then speed up again and liberate more potential energy, similar to a swinging pendulum.

We address these two problems by 1) evaluating only the highest 99th percentile of per-node kinetic energy against a threshold that we set as 10^{-8} and 2) ensuring that the threshold is met throughout a window of 0.1s of simulation.

Figure 7 (bottom) shows an example of early exit (bottom center), comparing the geometry at that point to the geometry obtained after an additional 0.5s of simulation, showing how all the draping details have settled and only slow, low frequency differences appear. The same scene without damping energies also reduces kinetic energy (due to the numerical damping introduced by Implicit Euler) but stays far from a low kinetic energy state and would require much more simulation time to reach the stop criterion.

6 Performance Features

In the previous section, we proposed a combination of parameter tuning and specific features that significantly reduced the overall computation time by efficiently guiding the simulation dynamics towards a converged state. In this section, we focus on a set of acceleration techniques that do not aim to influence the simulation dynamics nor the final drape beyond minor numerical differences, but to run computations significantly faster. These are *software optimization* techniques that do not require specialized knowledge on physics or cloth simulation and rely on the efficient implementation of numerical computation code.

6.1 Dual System Matrix

Each time the Newton integrator iterates, which is at least once per time step, the stiffness and damping matrices change and the entire system matrix (the left-hand side of Equation 4) needs to be rebuilt. Given the large number of degrees of freedom in the simulation, these matrices are too large to be dense and must be stored in a sparse format such as Compressed Sparse Row (CSR). Assembling a sparse matrix is a time consuming process and can account for a significant portion of the solver's computational budget.

Following prior work [12, 28], we note that the garment topology is constant throughout the simulation. With the notable exception of collisions, all energies of § 3.2 have static stencils. Therefore, we cast the system matrix into a sum of two matrices: one with static topology, which contains all static stencil energies, and one with dynamic topology which contains collision soft constraints.

Garment	Reassembled Matrix (s)	Dual Matrix (s)	Speedup
1)	12588	5196	2.42
2)	4442	864	5.14
3)	7414	1714	4.33
4)	9338	2503	3.73
5)	5335	1080	4.94
6)	5101	1150	4.44
7)	6184	1419	4.36
8)	18682	4460	4.19
9)	6902	1712	4.03
10)	9106	2113	4.31

Table 3: Computation times for simulations using a system matrix reassembled at each iteration and the Dual Matrix approach. The latter is much faster to compute, with speedups between 2.42x and 5.14x.

Garment	Single Precision (s)	Mixed Precision (s)	Double Precision (s)
1)	4055	4448	5196
2)	760	825	864
3)	1272	1499	1714
4)	2003	2074	2503
5)	958	970	1080
6)	1002	1068	1150
7)	1277	1298	1419
8)	3715	3942	4460
9)	1404	1439	1712
10)	1781	1848	2113

Table 4: Computation times for simulations using different numerical precision types. The mixed method achieves the best balance between precision and speed.

The matrix with static topology can be assembled only once and its values updated at each iteration, thus avoiding expensive matrix assembly operations. The dynamic matrix is reassembled at each iteration, but since it only contains entries related to collisions, it is much sparser and comparatively much faster to assemble than its static counterpart. Since we use Conjugate Gradient to solve Equation 4, we replace the single matrix-vector product per CG iteration with the sum of the two matrix-vector products corresponding to the static and dynamic matrices. The small overhead incurred by an additional, albeit small, matrix-vector product is largely offset by the computational savings of not having to assemble a large sparse matrix, as shown below.

To further reduce the density of the dynamic topology matrix, we only use it for off-diagonal entries. Entries in the 3x3 diagonal are guaranteed to exist in the static topology matrix, and we therefore accumulate them there. Similarly, mass entries are exclusively on the 3x3 diagonal and can therefore be accumulated into the static topology matrix.

Due to the compact nature of compressed sparse matrix formats, directly accessing the elements of the static topology matrix to update them is not cheap. Each direct access requires a search in linear time within an array that can have hundreds of entries. Again, since the topology is fixed, the mapping between energy stencil and sparse matrix entry is constant and can be pre-computed. We use this map to update the matrix entry in constant time. In addition, since each simulation node corresponds to 3 DoFs, we can store, access and update 3x3 blocks of entries instead of individual entries, which reduces the storage size of the stencil-to-matrix map and makes reads and writes more efficient.

Table 3 shows the computational cost of the dual matrix approach for the set of production garments, compared to reassembling the system matrix at each iteration, leading to very significant speedups ranging from 2.42x to 5.14x.

6.2 Single vs Double Precision

There is a clear trade-off between single and double precision data types. Using single precision (32 bits) allows for a more compact representation of floating-point numbers than double precision (64 bits). A smaller memory footprint results in a more efficient use of the memory, from transfer to caching. Particularly, modern compilers and off-the-shelf computation toolkits [23] increasingly use SIMD vectorization as an efficient low-level optimization technique to achieve significant speedups. Using single precision one can do twice as many floating-point operations in SIMD compared to double precision.

On the other hand, double precision types naturally have a much higher representation space regarding floating point numbers. Single precision types can quickly accumulate round-off errors in their least significant bits if multiple operations are involved. This is relevant for some energies such as bending, whose true-to-life stiffness values are extremely small, resulting in computations which struggle if the underlying data type does not have enough precision.

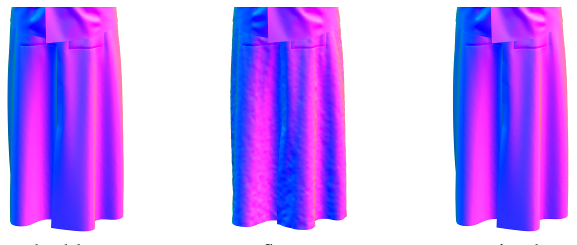


Fig. 8: Bottom part of the Autumn Outfit (8) after 3s of simulation with the three different precision modes. Note how the floats version shows noisy artifacts in the skirt region.

We explored the use of single precision, double precision, and a mix of both (as suggested by Kim and Eberle [28]) in terms of computation time. The mixed approach uses double precision when computing energy derivatives but single precision when solving the linear system from Equation 4. The results are shown in Table 4. As expected, single precision computations are faster for all garments. This is especially true during the linear solve step, since we use an off-the-shelf CG algorithm built with SIMD vectorization. However, when looking at the converged drapes of some garments we can clearly notice that the garment surface is noisy, as shown in Figure 8): we pay the price for the the lack of precision, and the results are of course unacceptable. The mixed approach seems to be the right compromise, as already hinted by Kim and Eberle. Computation times are close to the ones achieved with single precision, again thanks to the lesser memory footprint and the use of SIMD during the CG solve, but simulations do not suffer from a lack of precision and converge to valid states (Figure 8, right).

6.3 Preconditioning

Iterative methods for solving linear systems (such as Conjugate Gradient) usually benefit from system matrix preconditioning. Different preconditioners leverage different system matrix properties and provide different trade-offs between preconditioning quality and time required to compute the preconditioner itself [4, 71]. In practice, the gain in linear solve computation time must significantly outweigh the time required to apply the preconditioner.

Kim and Eberle [28] evaluated some of these preconditioners in a production environment and for a problem similar to ours and observed that a 3-Block Jacobi preconditioner consistently offers the best performance. We have tested this preconditioner, together with the Diagonal and the Incomplete Cholesky preconditioners, with our production scenes. The results, summarized in Figure 9, show that, while increasingly complex preconditioners can improve further the condition of the matrix and reduce the number of CG iterations, the performance gain is countered by the time required to apply the preconditioner itself. This is particularly evident for the Incomplete Cholesky preconditioner, which can reduce the average number of CG iterations

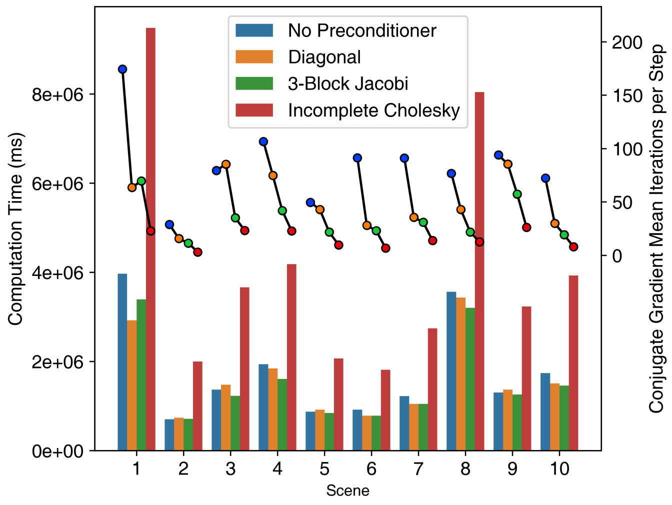


Fig. 9: Computation time (bars, left axis) and Conjugate Gradient mean iterations per time step (dots, right axis) using three different preconditioners and no preconditioner at all. Overall, 3-Block Jacobi offers the best speeds with an average 1.12x speedup, and Incomplete Cholesky the lesser number of Mean CG Iterations with an average 6.92x reduction.

by an impressive 83% w.r.t. the non-preconditioned case but results in an average 2.4x increase in computation time. The best performing preconditioner, aligning with the findings of Kim and Eberle, is the 3-Block Jacobi preconditioner, with an average speedup of 1.12x w.r.t the non-preconditioned case. We therefore use this preconditioner from this point on.

6.4 Degrees of Freedom Reordering

The cost of computing operations is, in general, sensitive to how data is laid out in memory. The Conjugate Gradient solve, and particularly its matrix-vector product, performs intensive computations that require many memory accesses to fetch both the values of the sparse matrix and the values of the dense vector. Data locality is known to increase the performance of this operation [53] for large enough systems where cache misses become an issue [64]. Since energy stencils represent mutual interactions between a closed set of DoFs, we can optimize for memory locality based on the stencil valence of each DoF. To this end, we cast the static topology stencils into an undirected graph where DoFs are graph nodes, and stencils are graph edges, effectively encoding the adjacency matrix of the DoFs. We then use existing algorithms that reduce the bandwidth of the graph, therefore optimizing the locality of the DoFs. In particular, and as suggested by Kim and Eberle [28], the *Reverse Cuthill-mcKee* is a fast and efficient graph bandwidth algorithm widely available in off-the-shelf graph processing libraries. We build the graph and run the algorithm once, during initialization. We obtain a permutation matrix that encodes the new DoF ordering, which we apply throughout our integrator. Figure 10 shows the typical sparsity pattern before and after the reordering, significantly reducing the matrix bandwidth [17].

Table 5 shows the computation times of the CG step, as well as the system matrix bandwidth, with and without DoF reordering. Interestingly, the impact is marginal in most cases, leading to CG computation speedups up to 1.15x in the best case, barely affecting the overall solver performance, as the linear system solve stages add up to $\sim$10% of the total time (Figure 11).

It is also worth noting that aside from improved memory access, reordering can also increase the effectiveness of some preconditioning strategies [53]. There is no gain in our case since the 3-Block Jacobi already operates on a dense 3-Block diagonal due to the 3D geometry of our simulations. However, it can become significant for problems of a different nature or a different choice of preconditioner.

Garment	Base (ms)	Reord. (ms)	CG Speedup	Base BW	Reord. BW
1)	264	251	1.05	38717	643
2)	22	23	0.98	27906	738
3)	87	84	1.03	46457	652
4)	124	125	1.00	30893	673
5)	40	38	1.03	33365	731
6)	36	35	1.03	28363	496
7)	65	56	1.15	38069	676
8)	122	122	1.00	39270	580
9)	124	121	1.02	41546	563
10)	62	63	0.98	45160	1202

Table 5: Experimental results of the reordering of DoFs, showing the average cost of the CG step (in ms) with and without DoF reordering, the resulting speedup in the CG solve and the system matrix bandwidth (BW) with and without reordering.

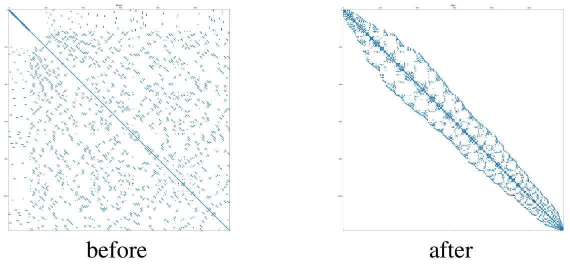


Fig. 10: Sparsity pattern of the system matrix of a garment, particularly coarse (1030 DoFs) for visualization purposes. Left: Sparsity pattern before reordering, with a matrix bandwidth of 380. Right: Sparsity pattern after reordering, with a matrix bandwidth of 59.

6.5 Stencil Coloring

Given the large number of energy stencils, computing energy derivatives in parallel is a natural way to achieve a significant speedup. For each force (or Hessian) computation, energy stencils can be distributed among threads, and each thread works through its assigned set of stencils. Each thread then accumulates the resulting forces (or Hessians) at a memory location corresponding to each degree of freedom involved in the stencil.

While this approach is simple to implement, it can lead to race conditions if two threads want to write on the memory location of a degree of freedom present in both stencils. In the case of Hessian computations and the dual system matrix from § 6.1, that memory location would be a 3x3 block of the static topology matrix. A straightforward solution is to use synchronization mechanisms, such as atomic operations, which guarantee that a given memory location is exclusive to one thread while the memory is being accessed, thus preventing concurrent updates. While this approach still leads to significant speedups, explicit synchronization mechanisms are costly due to the added overhead and the potential bottlenecks that, at worse, force threads to idle while waiting for their turn to access the memory.

Several strategies have been proposed to avoid explicit synchronization, such as a two-pass parallel accumulation approach [28] or stencil grouping based on graph coloring [19, 20]. We follow this latter approach and implement a stencil coloring mechanism that splits the stencils into groups so that the stencils in each group do not share any degrees of freedom. The stencils in each group are processed in parallel, while the groups themselves are processed sequentially.

We need to generate the groups only once, during initialization. For this, we cast the stencils and their degrees of freedom into an undirected graph, where the stencils are the nodes, and the degrees of freedom are the edges. We can then use a graph coloring algorithm to separate stencils into non-overlapping groups. Since our coloring is only computed once at the beginning of the simulation, coloring speed is not a constraint. We therefore use the algorithm by Coleman and

Garment	Synchronization (s)	Stencil Coloring (s)	Speedup
1)	3431	3113	1.10
2)	709	628	1.13
3)	1236	1112	1.11
4)	1606	1493	1.08
5)	848	759	1.12
6)	777	669	1.16
7)	1057	917	1.15
8)	3169	2874	1.10
9)	1239	1149	1.08
10)	1454	1342	1.08

Table 6: Computation times for simulations using explicit synchronization for energy and Hessian accumulation, and using coloring instead, showing an average speedup of 1.11x for the latter.

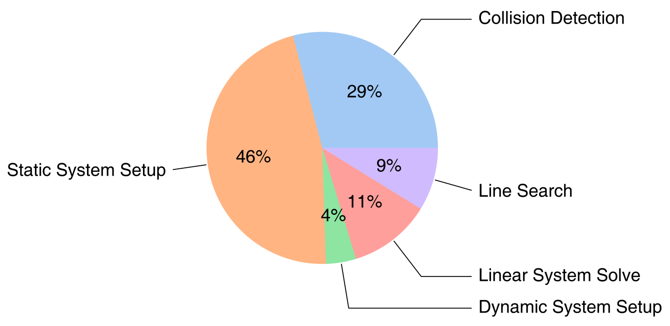


Fig. 11: Pie chart showing the relative time spent in each stage of the simulator. Static System Setup (structure initialization, energy accumulation, etc) and Collision Detection account for 75% of the total simulation time.

Moré [14] as it is readily available in off-the-shelf graph processing libraries and favors low color count rather than coloring speed. We refer the reader to the work by Fratarcangeli et al. [20] for an analysis of different heuristic coloring approaches when coloring speed is a constraint.

As a result, each stencil is labeled with a color index, and all stencils with the same index are guaranteed to have independent degrees of freedom and can therefore accumulate all force and Hessian contributions directly. Table 6 shows the acceleration introduced using stencil coloring compared to using synchronization via atomic operations for the accumulation of Hessian contributions and a two-pass approach [28] for force contributions. We note that the dynamic stencils generated by the collision detection system are not colored and are therefore processed in a serial manner. However, they are typically a small fraction of the total amount of processed stencils, involving simpler Hessian and force computations, and thus accounting for a small percentage of the total computation as shown in Figure 11.

7 Discussion

All the features we have presented and evaluated in this work increase the performance of our draping simulations without significantly affecting their quality.

The convergence features described in § 5 are largely specific to drape simulations. It is worth noting the impact of carefully choosing solver parameters, since very significant speed-ups can be obtained without significantly impacting the result.

The performance features described in § 6 are more generally applicable and are also capable of providing significant gains. Table 7 summarizes the cumulative contributions of the different performance features relative to the baseline and to each other. The net gain of all the applied features results in a significant speedup of 5.89x, with the Dual System Matrix as the largest contributor by far.

Figure 11 shows a breakdown of the average cost of a simulation step after applying all the features presented in this paper. One remarkable observation is that the system setup (mainly force and Hessian accumulations) takes significantly more time than the actual linear solve, even after the big improvement achieved in § 6.1. This contrasts with experimental results found in previous work dealing with similar problems [2]. However, it is not surprising when considering the differences in time stepping strategies. The smaller time step used in our case is favoring both the system to be better conditioned (as explained in § 3.1.3) and the initial inertial guess (§ 3.1) to be closer to the solution, leading to rapidly converging solves. This does not mean that a smaller time step is necessarily better. An excessively small time step can lead to a very inefficient solver, spending most of the resources setting up a problem and then barely advancing it.

Further discussion arises, then, regarding how larger time steps (and more generally, stiffer linear problems) could reshape the time distributions and how it could also change the impact of the evaluated features. Arguably, the net effect of the DoF reordering could be more significant, as there would be an increase in memory accesses during the linear solve. Similarly, the preconditioner could also gain in efficiency, maybe even leading to a different preferred choice: the Incomplete Cholesky computation time could pay off given the larger reduction in CG iterations.

Some of the features we presented (§ 6.1, 6.4, 6.5) require some precomputations that run before starting the simulation. The average precomputation time for the production scenes is 5.30s, which accounts for less than 1% of the total simulation time and can therefore be considered negligible.

In order to validate that our conclusions are not biased by the relatively small size of the garment dataset, and therefore generalize to a wider range of garments, we have curated and simulated a validation set made of 60 production garments. The simulation was done with the full set of convergence and performance features for the validation to remain computationally tractable. We computed the distribution of computation time, normalized by the number of degrees of freedom to make the metric comparable across garments. We then compared the results between the original 10 garments and the validation set, with the assumption that a similar qualitative distribution implies a similar overall quantitative effect of the features presented in this paper. Figure 12 shows indeed good agreement between both sets of garments, which is also true for the average early exit values (0.75s for the 10 original garments and 0.71s for the validation set), suggesting that our conclusions seem to generalize to a wider range of garments. We refer the reader to Appendix C in the supplementary material for additional information, including renders of the entire validation set.

7.1 Code Complexity

The complexity of implementing and maintaining an algorithm is an important factor to consider when developing any system. Having this information at hand, together with performance numbers, can help practitioners plan and prioritize the implementation of features. This diagnosis is typically neglected in academic literature, and for a good reason: it is hard to make an objective assessment. Personal assessments are subjective, influenced by skills and emotion. Code analysis tools are necessarily more objective but suffer from their generality: they likely fail to capture the intricacies specific to numerical and geometrical algorithms.

With this caveat in mind, but assuming code analysis tools are at least directionally correct, we use the Cognitive Complexity metric [11] already adopted by commercial code analysis suites. We compute the Cognitive Complexity of the implementation of each feature, excluding the routines of third-party libraries, and report the results in Table 7.

Interestingly, the Dual System Matrix (§ 6.1) provides the largest speedup but also has the largest cognitive complexity, in both cases by a wide margin. In practice, this feature requires changes that span the entire simulator: it is therefore advisable to implement it early on. On the other end, the Mixed Numerical Precision (§ 6.2) has a complexity of zero: a type alias and a few type casting calls. The remaining features (§ 6.3, § 6.4, § 6.5) have comparable moderate complexities. They

Metric	Dual Matrix	Mixed Numerical Precision	Preconditioning	DoF Reordering	Stencil Coloring
Speedup against previous column	4.19	1.13	1.12	1	1.11
Speedup against Baseline	4.19	4.73	5.30	5.30	5.89
Cognitive Complexity	227	0	31	23	20

Table 7: Speedup and Cognitive Complexity for all features related to § 6. Speedup is measured as the average between the production garments (Figures 1 and 4). The Dual System Matrix leads by a wide margin both in terms of speedup and complexity.

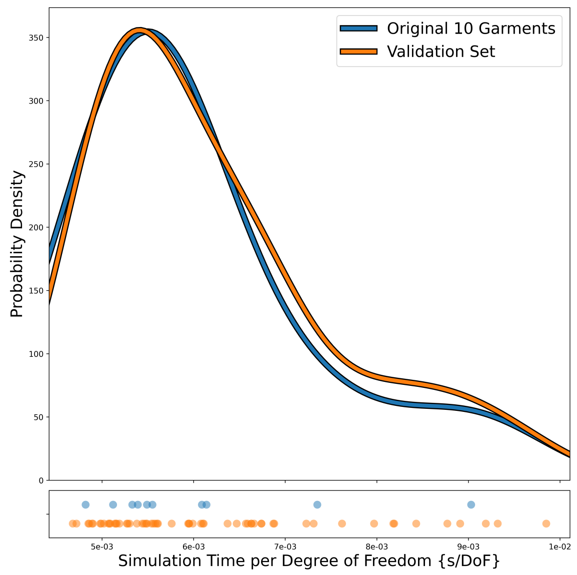


Fig. 12: The total drape simulation time per DoF for both the 10 original garments (blue) and the validation set (orange), plotted at the bottom. Their corresponding probability density functions, computed using Kernel Density Estimation (KDE), are plotted at the top. Both datasets yield very similar distributions, suggesting good generalization of the proposed features.

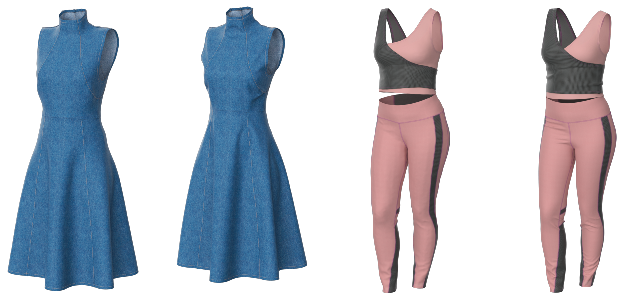


Fig. 13: Comparison of a dress and a sports outfit draped with the real time simulator (left) and our proposed simulator (right). The real-time draping of the dress is clearly overly elongated, but is accurately treated in our simulator. In the case of the sports outfit, characteristic subtle wrinkles emerge naturally when using an accurate simulator.

Scene	Nb DoFs	C-IPC (s)	Ours (s)
baseline	256353	2643	661
coarse	113763	1392	315
stiff stretch	256353	4443	710
compliant bending	256353	4512	722
stiff bending	256353	4910	721
2 stack	227526	9041	790
3 stack	341289	19670	1261

Table 8: Comparisons of wall-clock computation times between C-IPC and our simulator for 1 second of simulation. The *baseline* scene is a $1m^2$ fabric patch falling on a sphere as bundled with the C-IPC codebase. The *stiff stretch* scene uses stretch values 1 order of magnitude higher, reaching the range found in real fabrics. The *compliant bending* and *stiff bending* scenes use bending values 1 order of magnitude lower (resp. higher) for a more diverse drapability. The *coarse* scene has a lower discretization resolution, matching the one used for the garments in this paper. The *2 stack* and *3 stack* have respectively 2 and 3 fabric patches stacked on top of each other.

benefit from being very localized in the codebase, while relying on third-party toolkits for some of the heavy work.

7.2 Comparison with Other Approaches

In this section, we compare our simulator to existing solutions to highlight its benefits, both qualitatively and quantitatively. In particular, we compare our drapes to the ones generated by a real-time simulator based on PBD which, as argued in § 2, belongs to a family of techniques that trades accuracy for speed. We also compare the performance of our solver to C-IPC, a state of the art simulator similar to ours in that it is a *global* method, thus prioritizing accuracy over speed.

PBD: Figure 13 shows side-by-side comparisons of the drapes from this simulator and a real-time simulator based in PBD. Both simulators use the same energy formulation and the same timestep size, with a PBD-style treatment for the latter [3] using 15 projection iterations. Mechanical parameters for both solvers were derived from the same captured data. We used the real-time simulator to assemble and arrange all the garments in this paper, using a discretization of 1cm which is standard in CAD for apparel [10, 18]. Qualitatively comparing the drapes between both simulators provides evidence of how a global simulator can dramatically improve the accuracy of the resulting drape (albeit at the cost of speed). In general, the PBD simulator is able to match the real behavior for small fabric samples, and its interactivity and stability capabilities are crucial for the design stage. However, for larger garments the behavior starts to deviate, as shown in the figure.

C-IPC: We then compared our simulator and C-IPC, a state of the art *global* simulator whose code is open source. We used the C-IPC simulator as is, using a bundled scene as the starting point where a square patch of fabric falls onto a sphere. We reproduced the same scene in our simulator, with the same positions, the same discretization resolution and hand-tuned parameters to try to achieve some similarity with the C-IPC drapings, which is particularly difficult given the difference in constitutive models, collision mechanisms, friction parametrizations, damping behavior, etc. We then created new scenes by altering some fundamental features of the original scene, in order to test different conditions: a lower resolution to match the resolution used in our simulator, a fabric with stiffer *stretch* parameters to match the behavior of real fabrics, fabrics with either stiffer or more compliant *bending* parameters for a wider drapability range, and additional fabric patches to mimic multi-layer garment constructions. Table 8 compares the performance numbers between both simulators for all scenes, and side-by-side renders are provided in Appendix D in the supplementary material, together with the respective material parameters. Both simulators reach interpenetration-free drapes for all scenes. C-IPC has the notorious advantage of allowing smaller contacts distances than in any discrete approach, which could lead to better handling of multiple stacked contacts. However, it is significantly slower than our simulator (from 4x to 15x depending on the scene) despite using timesteps up to 0.04s, and even prohibitively slow in some cases for production pipelines.

8 Conclusions

In this paper, we have discussed and evaluated a set of features to improve the performance of a production Newton-Backward Euler draping simulator without sacrificing its accuracy. We have introduced a set of convergence features that alter the dynamics to produce a faster drape, and a set of performance features that speed up numerical computations. We have implemented and evaluated all features experimentally using a set of production scenes with realistic materials. We have then shown that relatively simple changes, such as the careful tuning of solver parameters and the separate treatment of static and dynamic topology energies can lead to significant performance benefits.

While our simulator has successfully simulated thousands of production scenes, it is not without its limitations, both in terms of robustness and of missed performance opportunities.

Our discrete collisions approach can handle virtually all configurations we have found in production. However, it relies on the use of a small time step and the fact that drape simulations do not typically involve fast moving elements. Therefore, it provides no theoretical guarantees of either resolving pre-existing penetrations from the input mesh or preventing new ones from appearing. We would like to explore the use of more theoretically robust approaches, such as Incremental Potential Contact [33,34], together with untangling mechanisms [1,28,66] for those bad initial configurations. While more computationally expensive, these techniques would provide better guarantees and likely allow us to safely increase the time step of the simulations.

We have refrained from exploring low level optimizations, such as explicit SIMD implementations or hardware-specific, cache-friendly data structures. We would like to explore ways of introducing these in a production environment without ending up with a system that is hard to extend, maintain and debug. According to Figure 11, force and Hessian computations seem to be a good place to start. We have also focused on exploiting the capabilities of modern CPU hardware, as it is a cheaper commodity in cloud computing environments used in production. However, we expect cloud GPU computing to become a more accessible commodity in the near future. A lot of research has already proven that great benefits come from running full simulations on the GPU [13,62], including GPU-friendly preconditioning strategies [69, 72] and collision detection and resolution [30, 32, 63]. We will explore the migration of our production solver to a massively parallel GPU implementation.

Finally, since our simulator is integrating the equations of motion, a few adjustments can enable it to simulate garments in motion, including animated avatars and interaction with other scene elements.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful suggestions, Harrison Johnson and Loreto Pérez for their garment designs and the SEDDI teams in general for their help and support. This work was funded in part by the Spanish Ministry of Science, Innovation and Universities (MCIN/AEI/10.13039/501100011033) and the European Union NextGenerationEU/PRTR programs through the TaiLOR project (CPP2021-008842). Dr. José María Pizana was partially supported by an Industrial Doctorate fellowship (DIN2019-010912).

References

[1] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. ACM Transactions on Graphics (TOG), 22(3):862–870, 2003. 13

[2] D. Baraff and A. P. Witkin. Large steps in cloth simulation. Proceedings of the 25th annual conference on Computer graphics and interactive techniques, 1998. 2, 4, 8, 11

[3] J. Bender, M. Müller, and M. Macklin. Position-Based Simulation Methods in Computer Graphics. In M. Zwicker and C. Soler, eds., Eurographics 2015 - Tutorials. The Eurographics Association, 2015. doi: 10.2312/egt.20151045 3, 12

[4] M. Benzi. Preconditioning techniques for large linear systems: a survey. Journal of computational Physics, 182(2):418–477, 2002. 9

[5] A. Berg, S. Hedrich, T. Lange, K.-H. Magnus, and B. Mathews. The apparel sourcing caravan's next stop: Digitization, 2017. 1

[6] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. Projective dynamics: Fusing constraint projections for fast simulation. ACM Trans. Graph., 33(4), jul 2014. doi: 10.1145/2601097.2601116 2

[7] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 594–603, 2002. 4

[8] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 01 2003. doi: 10.1145/1198555.1198573 4

[9] G. E. Brown, M. Overby, Z. Forootaninia, and R. Narain. Accurate dissipative forces in optimization integrators. ACM Transactions on Graphics (TOG), 37(6):1–14, 2018. 8

[10] Browzwear. VStitcher. https://browzwear.com/products/v-stitcher, 2023. 3, 12

[11] G. A. Campbell. Cognitive complexity: An overview and evaluation. In Proceedings of the 2018 international conference on technical debt, pp. 57–58, 2018. 11

[12] G. Cirio, J. Lopez-Moreno, D. Miraut, and M. A. Otaduy. Yarn-level simulation of woven cloth. ACM Trans. on Graphics (Proc. of ACM SIGGRAPH Asia), 33(6), 2014. 8

[13] G. Cirio, J. Lopez-Moreno, and M. A. Otaduy. Yarn-level cloth simulation with sliding persistent contacts. IEEE transactions on visualization and computer graphics, 23(2):1152–1162, 2016. 13

[14] T. F. Coleman and J. J. More. Estimation of sparse jacobian matrices and graph coloring problems. Journal of Numerical Analysis, 20:187–209, 1983. 11

[15] N. Corporation. NVIDIA PhysX SDK. https://github.com/NVIDIA-Omniverse/PhysX, December 2018. 3

[16] E. Coumans and Y. Bai. Bullet Physics SDK. http://bulletphysics.org/, 2016–2023. 3

[17] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In Proceedings of the 1969 24th National Conference, ACM '69, p. 157–172. Association for Computing Machinery, New York, NY, USA, 1969. doi: 10.1145/800195.805928 10

[18] C. V. Fashion. Clo. https://www.clo3d.com/es/clo, 2023. 3, 12

[19] M. Fratarcangeli and F. Pellacini. Scalable partitioning for parallel position based dynamics. In Computer Graphics Forum, vol. 34, pp. 405–413. Wiley Online Library, 2015. 10

[20] M. Fratarcangeli, V. Tibaldo, and F. Pellacini. Vivace: A practical gauss-seidel method for stable soft body dynamics. ACM Transactions on Graphics (TOG), 35(6):1–9, 2016. 2, 10, 11

[21] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran. Optimization integrator for large time steps. IEEE transactions on visualization and computer graphics, 21(10):1103–1115, 2015. 2, 4, 6

[22] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '03, p. 62–67. Eurographics Association, Goslar, DEU, 2003. 4

[23] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. 4, 9

[24] H. Han, M. Sun, S. Zhang, D. Liu, and T. Liu. Gpu cloth simulation pipeline in lightchaser animation studio. In SIGGRAPH Asia 2021 Technical Communications, SA '21 Technical Communications. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3478512.3488616 3

[25] D. Harmon, E. Vouga, R. Tamstorf, and E. Grinspun. Robust treatment of simultaneous collisions. In ACM SIGGRAPH 2008 papers, pp. 1–4. 2008. 4

[26] G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs. An implicit finite element method for elastic solids in contact. In Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No. 01TH8596), pp. 136–254. IEEE, 2001. 4

[27] V. Hämmerle, C. Mühlenbein, M. Rüßmann, C. Gauger, and S. Rohrhofer. Why fashion must go digital—end to end, 2020. 1

[28] T. Kim and D. Eberle. Dynamic deformables: Implementation and production practicalities (now with code!). In ACM SIGGRAPH 2022 Courses, SIGGRAPH '22. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3532720.3535628 3, 5, 8, 9, 10, 11, 13

[29] T. Kim and D. Eberle. Hobak: A Library for Squashing Things. https://github.com/theodorekim/HOBAKv1, 2022. 3

[30] L. Lan, G. Ma, Y. Yang, C. Zheng, M. Li, and C. Jiang. Penetration-free projective dynamics on the gpu. ACM Transactions on Graphics (TOG), 41(4):1–16, 2022. 13

[31] S. Lesser, A. Stomakhin, G. Daviet, J. Wretborn, J. Edholm, N.-H. Lee, E. Schweickart, X. Zhai, S. Flynn, and A. Moffat. Loki: A unified multiphysics simulation framework for production. ACM Trans. Graph., 41(4), jul 2022. doi: 10.1145/3528223.3530058 3

[32] C. Li, M. Tang, R. Tong, M. Cai, J. Zhao, and D. Manocha. P-cloth: interactive complex cloth simulation on multi-gpu systems using dynamic matrix assembly and pipelined implicit integrators. ACM Transactions on Graphics (TOG), 39(6):1–15, 2020. 13

[33] M. Li, Z. Ferguson, T. Schneider, T. R. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. ACM Trans. Graph., 39(4):49, 2020. 4, 5, 13

[34] M. Li, D. M. Kaufman, and C. Jiang. Codimensional incremental potential contact. ACM Trans. Graph., 40(4), 2021. doi: 10.1145/3450626.3459767 3, 5, 13

[35] X. Li, Y. Fang, L. Lan, H. Wang, Y. Yang, M. Li, and C. Jiang. Subspace-preconditioned gpu projective dynamics with contact for cloth simulation. In SIGGRAPH Asia 2023 Conference Papers, pp. 1–12, 2023. 2

[36] C. Liu, W. Xu, Y. Yang, and H. Wang. Automatic digital garment initialization from sewing patterns. ACM Trans. Graph. (SIGGRAPH), jul 2024. 7

[37] T. Liu, S. Bouaziz, and L. Kavan. Quasi-newton methods for real-time simulation of hyperelastic materials. ACM Trans. Graph., 36(4), jul 2017. doi: 10.1145/3072959.2990496 2

[38] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and T.-Y. Kim. Primal/dual descent methods for dynamics. In Computer Graphics Forum, vol. 39, pp. 89–100. Wiley Online Library, 2020. 2, 3

[39] M. Macklin, M. Müller, and N. Chentanez. Xpbd: Position-based simulation of compliant constrained dynamics. In Proceedings of the 9th International Conference on Motion in Games, MIG '16, p. 49–54. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2994258.2994272 3

[40] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim. Unified particle physics for real-time applications. ACM Transactions on Graphics (TOG), 33(4):104, 2014. 3

[41] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross. Example-based elastic materials. ACM Transactions on Graphics (TOG), 30(4):1–8, 2011. 4

[42] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis. Efficient elasticity for character skinning with contact and collisions. In ACM SIGGRAPH 2011 papers, pp. 1–12. 2011. 4

[43] E. Miguel, R. Tamstorf, D. Bradley, S. C. Schvartzman, B. Thomaszewski, B. Bickel, W. Matusik, S. Marschner, and M. A. Otaduy. Modeling and estimation of internal friction in cloth. ACM Transactions on Graphics (TOG), 32(6):1–10, 2013. 4

[44] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. J. Vis. Comun. Image Represent., 18(2):109–118, apr 2007. doi: 10.1016/j.jvcir.2007.01.005 3

[45] K. Museth, N. Avramoussis, and D. Bailey. Openvdb. In ACM SIGGRAPH 2019 Courses, pp. 1–56. 2019. 4

[46] M. Müller. Hierarchical Position Based Dynamics. In F. Faure and M. Teschner, eds., Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008). The Eurographics Association, 2008. doi: 10.2312/PE/vriphys/vriphys08/001-010 3

[47] R. Narain, A. Samii, and J. F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. ACM Transactions on Graphics, 31(6):147:1–10, Nov. 2012. Proceedings of SIGGRAPH Asia. 3

[48] J. Nocedal and S. J. Wright. Numerical Optimization. Springer, New York, NY, USA, 2e ed., 2006. 2, 4

[49] Optitex. Pattern Design Software 3D. https://optitex.com/es/products/2d-and-3d-cad-software/, 2023. 3

[50] M. Overby, G. E. Brown, J. Li, and R. Narain. Admm $\supseteq$ projective dynamics: Fast simulation of hyperelastic models with dynamic constraints. IEEE Transactions on Visualization and Computer Graphics, 23(10):2222–2234, 2017. doi: 10.1109/TVCG.2017.2730875 2

[51] J. W. S. B. Rayleigh. The theory of sound, vol. 2. Macmillan, 1896. 8

[52] B. Roberts-Islam. Is digitization the savior of the fashion industry?, 2020. 1

[53] S. P. Sastry, E. Kultursay, S. M. Shontz, and M. T. Kandemir. Improved cache utilization and preconditioner efficiency through use of a space-filling curve mesh element-and vertex-reordering technique. Engineering with Computers, 30:535–547, 2014. 10

[54] SideFX. Houdini. https://www.sidefx.com/products/houdini/, 2023. 3

[55] C. Soler, T. Martin, and O. Sorkine-Hornung. Cosserat rods with projective dynamics. In Computer Graphics Forum, vol. 37, pp. 137–147. Wiley Online Library, 2018. 3

[56] G. Sperl, R. Narain, and C. Wojtan. Homogenized yarn-level cloth. ACM Transactions on Graphics (TOG), 39(4):48–1, 2020. 5

[57] J. Stam. Nucleus: Towards a unified dynamics solver for computer graphics. pp. 1 – 11, 09 2009. doi: 10.1109/CADCG.2009.5246818 3

[58] R. M. Sánchez-Banderas and M. A. Otaduy. Strain rate dissipation for elastic deformations. Computer Graphics Forum (Proc. of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation), 37(8):161–170, 2018. 4, 8

[59] R. Tamstorf and E. Grinspun. Discrete Bending Forces and Their Jacobians. Graph. Models, 75(6):362–370, Nov. 2013. doi: 10.1016/j.gmod.2013.07.001 4

[60] R. Tamstorf, T. Jones, and S. F. McCormick. Smoothed aggregation multigrid for cloth simulation. ACM Trans. Graph., 34(6), nov 2015. doi: 10.1145/2816795.2818081 3, 4

[61] M. Tang, R. Tong, Z. Wang, and D. Manocha. Fast and exact continuous collision detection with bernstein sign classification. ACM Transactions on Graphics (TOG), 33(6):1–8, 2014. 5

[62] M. Tang, H. Wang, L. Tang, R. Tong, and D. Manocha. Cama: Contact-aware matrix assembly with unified collision handling for gpu-based cloth simulation. In Computer Graphics Forum, vol. 35, pp. 511–521. Wiley Online Library, 2016. 4, 13

[63] M. Tang, T. Wang, Z. Liu, R. Tong, and D. Manocha. I-cloth: Incremental collision handling for gpu-based interactive cloth simulation. ACM Transactions on Graphics (TOG), 37(6):1–10, 2018. 4, 13

[64] S. Toledo. Improving the memory-system performance of sparse-matrix vector multiplication. IBM Journal of research and development, 41(6):711–725, 1997. 10

[65] T. Trusty, D. Kaufman, and D. I. Levin. Mixed variational finite elements for implicit simulation of deformables. In SIGGRAPH Asia 2022 Conference Papers, pp. 1–8, 2022. 6

[66] P. Volino and N. Magnenat-Thalmann. Resolving surface collisions through intersection contour minimization. ACM Transactions on Graphics (TOG), 25(3):1154–1159, 2006. 13

[67] P. Volino, N. Thalmann, and F. Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. ACM Transactions on Graphics, 28, 08 2009. doi: 10.1145/1559755.1559762 4

[68] H. Wang. Defending continuous collision detection against errors. ACM Transactions on Graphics (TOG), 33(4):1–10, 2014. 5

[69] H. Wang and Y. Yang. Descent methods for elastic body simulation on the gpu. ACM Transactions on Graphics (TOG), 35(6):1–10, 2016. 13

[70] Z. Wang, M. Tang, R. Tong, and D. Manocha. Tightccd: Efficient and robust continuous collision detection using tight error bounds. In Computer Graphics Forum, vol. 34, pp. 289–298. Wiley Online Library, 2015. 5

[71] A. J. Wathen. Preconditioning. Acta Numerica, 24:329–376, 2015. 9

[72] B. Wu, Z. Wang, and H. Wang. A gpu-based multilevel additive schwarz preconditioner for cloth and deformable body simulation. ACM Transactions on Graphics (TOG), 41(4):1–14, 2022. 13

[73] Z. Xian, X. Tong, and T. Liu. A scalable galerkin multigrid method for real-time simulation of deformable objects. ACM Transactions on Graphics (TOG), 38(6):1–13, 2019. 2

[74] H. Xu and J. Barbič. Example-based damping design. ACM Transactions on Graphics (TOG), 36(4):1–14, 2017. 8

[75] K. Yamane and Y. Nakamura. Stable penalty-based model of frictional contacts. In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pp. 1904–1909. IEEE, 2006. 4

[76] C. Yuksel, J. M. Kaldor, D. L. James, and S. Marschner. Stitch meshes for modeling knitted clothing with yarn-level detail. ACM Transactions on Graphics (TOG), 31(4):1–12, 2012. 7

[77] J. E. Zhang, J. Dumas, Y. Fei, A. Jacobson, D. L. James, and D. M. Kaufman. Progressive simulation for cloth quasistatics. ACM Transactions on Graphics (TOG), 41(6):1–16, 2022. 7